Ⅸ. 2 次元拡散方程式のスレッド並列チューニングについて

富士通株式会社
谷村 恭伸

この報告は、「2.7 スレッド並列プログラムの性能測定」内の「３．2 次元拡散方程式」のチューニング初期段階において発生した性能問題に関しての分析結果をまとめたものです。なお、この版には、最終版に存在しない暫定的に記述したデータコピー処理が含まれているため、それが影響して発生した問題も含まれています。

指摘された性能問題は、表１の測定値における以下の３点です。
(1) 10 並列から 20 並列で、スケーラビリティが 2 倍以上出ている
(2) 案 1，2，3 の 60 並列が遅い
(3) 案 1 と案 2 の 20 並列が遅い
表中のプログラムは、それぞれ図１～図４に対応します。案１～案３のチューニングは、オリジナルの B の部分のコストが大きいので、これを解消するために A と B の部分を１つにまとめたチューニングを何パターンか試されたものです。

表１

	案１	案２	案３	オリジナル (A)	オリジナル (B)
1 並列	42.13	42.07	41.65	42.07	42.07
10 並列	7.30	7.29	7.12	6.20	6.20
20 並列	1.75	1.79	0.70	0.67	0.67
30 並列	0.69	0.80	0.62	0.46	0.46
40 並列	0.41	0.46	0.38	0.35	0.35
50 並列	0.35	0.40	0.33	0.30	0.30
60 並列	0.73	0.85	0.62	0.26	0.26

1. 10 並列から 20 並列で、スケーラビリティが 2 倍以上出ている原因について

永井委員の予想どおり、20 並列からキャッシュに乗り切るようになるためです。
対象ループで使用されているデータの大きさは、約 60M バイト(2001 要素*2001 要素*8 バイト*2 個)です。並列実行時に各 CPU で扱うデータ量は、以下のようになります。
10 並列：約 6M バイト
20 並列：約 3M バイト
したがって、L2 キャッシュの４M バイトに乗り切ることになります。また、プロファイラにより L2 キャッシュミスが軽減されることも確認済みです。

2. 案 1, 2, 3 の 60 並列が遅い原因について

キャッシュミスが発生するために、パラレルバランスが悪くなっていることが原因です。
同期待ち時間を除いた各スレッドの並列実行時間（パラレルバランス）を確認したところ、各スレッドの実行時間の幅は以下のとおりでした。
50 並列：約 50μ秒
60 並列：約 400μ秒
表１の 50 並列と 60 並列の時間の差分から、これが低下要因であるということが分かります。原因は以下のとおりです。

チューニング版では、サブルーチン diffuse の先頭でのデータコピーのループ(X)が、0～jmax で並列化されているのに対して、測定区間のループ(Y)では、1～jmax-1 で並列化されています。そのため、ループ X とループ Y で、各 CPU に割り当てられる要素の端にずれが生じます。これが、キャッシュミスの要因となり、高並列度での影響が大きくなるためです。しかし、オリジナル版では、配列 wk は 1～jmax-1 でしか書き込みされないので、各 CPU に割り当てられる要素にずれは生じることはありません。

3. 案 1 と案 2 の 20 並列が遅い原因について

コンパイラの最適化状況に違いがあるためです。

これは、定義と参照が同一配列の場合（案 1，2）に、コンパイラでデータ依存関係を見切れないため、定義と参照が別配列の場合（案 3、オリジナル）には動作するループの回転を跨った最適化がされていないためです。今後、コンパイラを改善する方向で検討いたします。

図１　オリジナル（定義と参照で依存があるため、ワーク配列を使用）

```
      parameter (imax=2000,jmax=2000)
      implicit real*8 (a-h,o-z)
      dimension u(0:imax,0:jmax),wk(0:imax,0:jmax)

c ---- step-A ----------------------
      call gettod()
      do j=1,jmax-1
         do i=1,imax-1
            wk(i,j)=sx*u(i-1,j)+(1.0d0-2.0d0*sx-2.0d0*sy)*u(i,j)
     *          +sx*u(i+1,j)+sy*u(i,j-1)+sy*u(i,j+1)
         enddo
       enddo
       call gettod()
c ---- step-B ----------------------
       call gettod()
       do j=1,jmax-1
          do i=1,imax-1
             u(i,j)=wk(i,j)
          enddo
       enddo
       call gettod()
```

図２　案１（3 次元にして依存をなくした(ocl で指示)）

```
       dimension u(0:imax,0:jmax,2)

       if(kk.eq.1) then
          k1=1
          k2=2
       else
          k1=2
          k2=1
       endif

       do j=0,jmax
          do i=0,imax
             u(i,j,k1)=v(i,j)
          enddo
       enddo

       call gettod()
!ocl norecurrence(u)
       do j=1,jmax-1
          do i=1,imax-1
             u(i,j,k2)=sx*u(i-1,j,k1)+(1.0d0-2.0d0*sx-2.0d0*sy)*u(i,j,k1)
     *          +sx*u(i+1,j,k1)+sy*u(i,j-1,k1)+sy*u(i,j+1,k1)
          enddo
       enddo

       call gettod()

       do j=0,jmax
          do i=0,imax
             v(i,j)=u(i,j,k2)
          enddo
       enddo
```

図３　案２（案１を、ocl 指定なしにできるように if で分けた）

```
      dimension v(0:imax,0:jmax,2)

      k1=mod(k-1,2)+1

      do j=0,jmax
        do i=0,imax
          v(i,j,k1)=u(i,j)
        enddo
      enddo

      call gettod()
      if(k1.eq.1) then
      do j=1,jmax-1
        do i=1,imax-1
          v(i,j,k1+1)=sx*v(i-1,j,k1)+(1.0d0-2.0d0*sx-2.0d0*sy)*v(i,j,k1)
     *          +sx*v(i+1,j,k1)+sy*v(i,j-1,k1)+sy*v(i,j+1,k1)
        enddo
      enddo
      else
      do j=1,jmax-1
        do i=1,imax-1
          v(i,j,k1-1)=sx*v(i-1,j,k1)+(1.0d0-2.0d0*sx-2.0d0*sy)*v(i,j,k1)
     *          +sx*v(i+1,j,k1)+sy*v(i,j-1,k1)+sy*v(i,j+1,k1)
        enddo
      enddo
      call gettod()

      if(k1.eq.1) then
      do j=0,jmax
        do i=0,imax
          u(i,j)=v(i,j,k1+1)
        enddo
      enddo
      else
      do j=0,jmax
        do i=0,imax
          u(i,j)=v(i,j,k1-1)
        enddo
      enddo
      endif
```

図４　案３（案２の３次元配列を、２つの２次元配列に分けた）

```
      dimension u1(0:imax,0:jmax),u2(0:imax,0:jmax)

      k1=mod(k,2)

      if(k1.ne.1) then
      do j=0,jmax
        do i=0,imax
          u2(i,j)=u1(i,j)
        enddo
      enddo
      endif

      call gettod()
      if(k1.eq.1) then
      do j=1,jmax-1
        do i=1,imax-1
          u2(i,j)=sx*u1(i-1,j)+(1.0d0-2.0d0*sx-2.0d0*sy)*u1(i,j)
     *          +sx*u1(i+1,j)+sy*u1(i,j-1)+sy*u1(i,j+1)
        enddo
      enddo
      else
      do j=1,jmax-1
        do i=1,imax-1
          u1(i,j)=sx*u2(i-1,j)+(1.0d0-2.0d0*sx-2.0d0*sy)*u2(i,j)
     *          +sx*u2(i+1,j)+sy*u2(i,j-1)+sy*u2(i,j+1)
        enddo
      enddo
      endif
      call gettod()

      if(k1.eq.1) then
      do j=0,jmax
        do i=0,imax
          u1(i,j)=u2(i,j)
        enddo
      enddo
      endif
```

- 以上 -